NNFL

Lecture 6

Different Learning Methods

1 **Supervised learning**, as the name indicates, has the **presence of a supervisor as a teacher**.

- Basically supervised learning is when we teach or train the machine using data that is well labeled.
- Which means some **data is already tagged with the correct answer**.
- After that, the machine is provided with a new set of examples(data) so that the supervised learning algorithm analyses the test and produces a correct labele.
-

1 **Supervised learning**

- **For instance**, suppose you are given a basket filled with different kinds of fruits. Now the first step is to train the machine with all different fruits one by one like this:

1 **Supervised learning**

- If the shape of the object is rounded and has a depression at the top, is red in color, then it will be labeled as –**Apple**.
- If the shape of the object is a long curving cylinder having Green-Yellow color, then it will be labeled as –**Banana**.

1 **Supervised learning**

- Now suppose after training the data, you have given a new separate fruit, say Banana from the basket, and asked to identify it.
- Since the machine has already learned the things from previous data and this time have to use it wisely.
- It will first classify the fruit with its shape and color and would confirm the fruit name as BANANA and put it in the Banana category.
- Thus the machine learns the things from training data(basket containing fruits) and then applies the knowledge to test data(new fruit).

1 Supervised learning

- **Classification**: A classification problem is when the output variable is a category, such as “Red” or “blue” or “disease” and “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.
- Supervised learning deals with or learns with “labeled” data. This implies that some data is already tagged with the correct answer.
- **Types:-**
- Regression
- Logistic Regression
- Classification
- Naive Bayes Classifiers
- K-NN (k nearest neighbors)
- Decision Trees
- Support Vector Machine

1 **Supervised learning**

- **Advantages:-**
- Supervised learning allows collecting data and produces data output from previous experiences.
- Helps to optimize performance criteria with the help of experience.
- Supervised machine learning helps to solve various types of real-world computation problems.

1 **Supervised learning**

- **Disadvantages:-**
- Classifying big data can be challenging.
- Training for supervised learning needs a lot of computation time. So, it requires a lot of time.

Supervised learning Real life Examples

- Train **your handwriting to OCR system** and once trained, it will be able to convert your hand-writing images into text (till some accuracy obviously)
- Based on some prior knowledge (when its sunny, **temperature is higher**; when its cloudy, **humidity is higher**, etc.) weather apps predict the parameters for a given time.
- Based on past information about spams, **filtering out a new incoming email** into **Inbox** (normal) or **Junk folder** (Spam)
- **Biometric attendance or ATM etc systems** where you train the machine after couple of inputs (**of your biometric identity - be it thumb or iris, etc.**), machine can validate your future input and identify you.

2 Unsupervised learning

- Unsupervised learning is the training of a machine using information that is neither classified nor labeled and allowing the algorithm to act on that information without guidance.
- Here the task of the machine is to **group unsorted information** according to similarities, patterns, and differences **without any prior training of data.**

2 Unsupervised learning

- Unlike supervised learning, no teacher is provided
- Therefore the machine is restricted **to find the hidden structure in unlabeled data by itself**.

2 Unsupervised learning

- **For instance**, suppose given images having both dogs and cats which it has never seen.

2 Unsupervised learning

- Thus the machine has **no idea about the features** of dogs and cats so it can't categorize it as 'dogs and cats '.
- But it **can categorize them according** to their similarities, patterns, and differences,
- i.e., it can easily categorize the above picture into two parts. The first may contain all pics having **dogs** in it and
- the second part may contain all pics having **cats** in it.
- It allows the model **to work on its own** to **discover patterns and information that was previously undetected**.
- It mainly **deals with unlabelled data.**

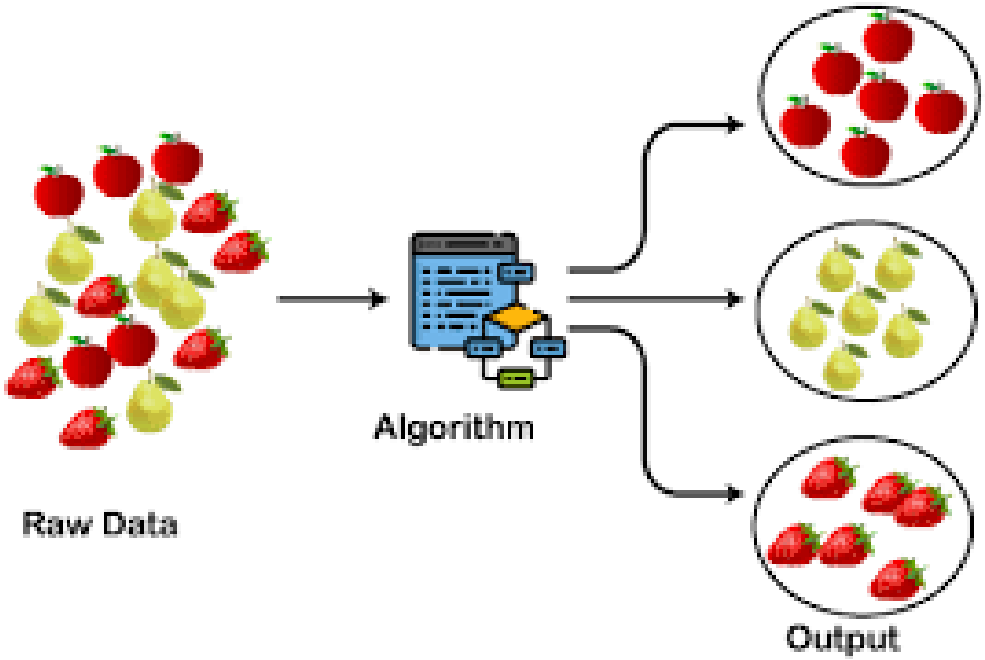
2 Unsupervised learning

- **Unsupervised learning is classified into two categories of algorithms:**

  Clustering: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.

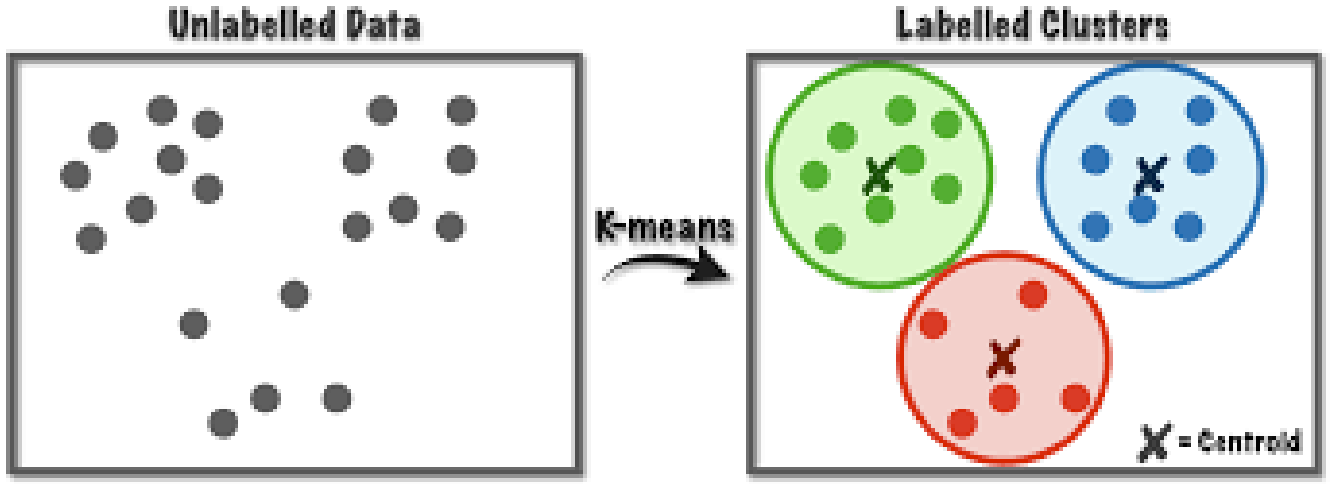
2 Unsupervised learning

- **Clustering**



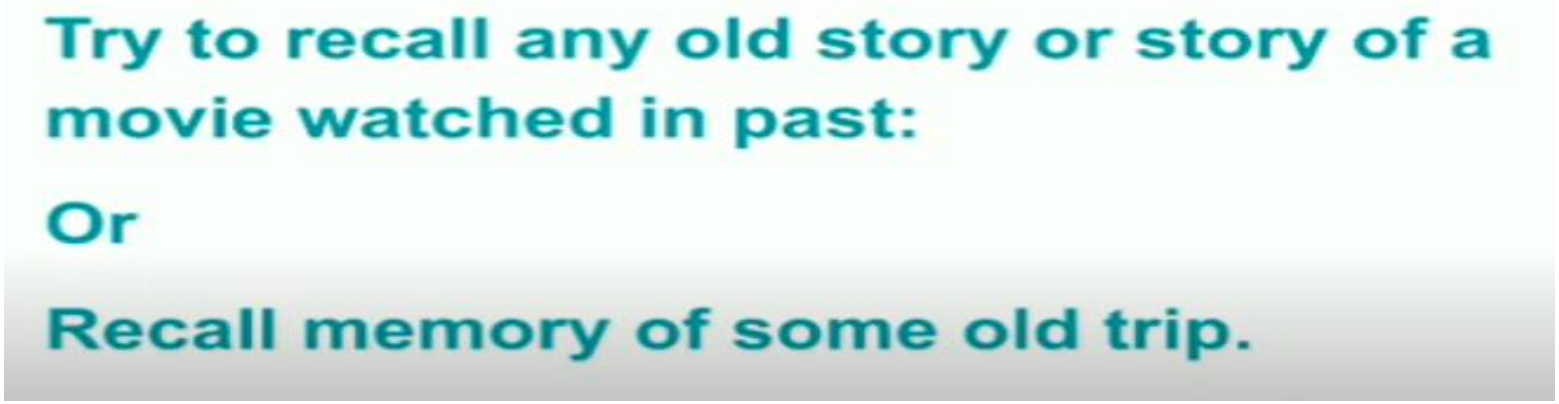
2 Unsupervised learning

- **Clustering**



2 Unsupervised learning

- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.



2 Unsupervised learning

Types of Unsupervised Learning:-

- **Clustering**
- Exclusive (partitioning)
- Agglomerative
- Overlapping
- Probabilistic
- **Clustering Types:-**
- Hierarchical clustering
- K-means clustering
- Principal Component Analysis
- Singular Value Decomposition
- Independent Component Analysis

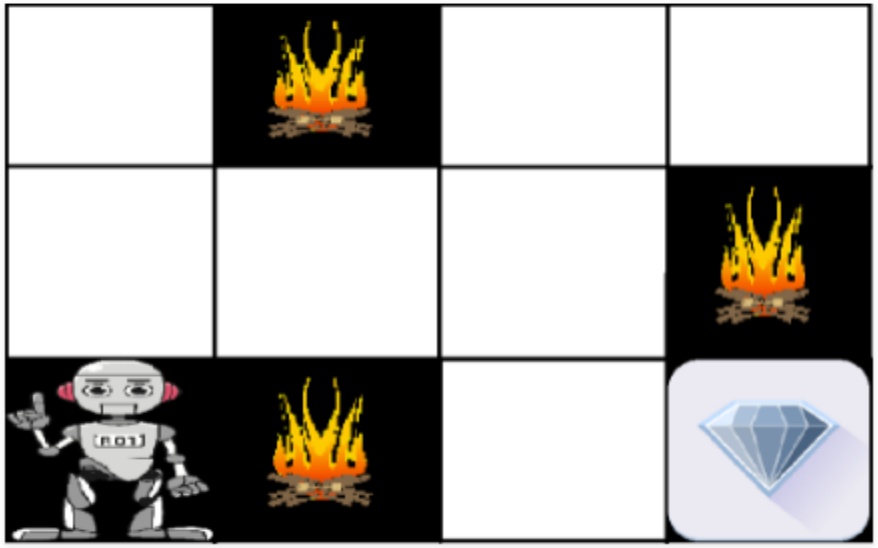
Supervised vs. Unsupervised Machine Learning

Parameters	Supervised machine learning	Unsupervised machine learning
Input Data	Algorithms are trained using labeled data.	Algorithms are used against data that is not labeled
Computational Complexity	Simpler method	Computationally complex
Accuracy	Highly accurate	Less accurate

Ref:https://www.geeksforgeeks.org/supervised-unsupervised-learning/

3 Reinforcement learning

- Reinforcement learning is about taking suitable action to **maximize reward in a particular situation**.
- It is employed by various software and machines **to find the best possible behavior or path it should take in a specific situation.**

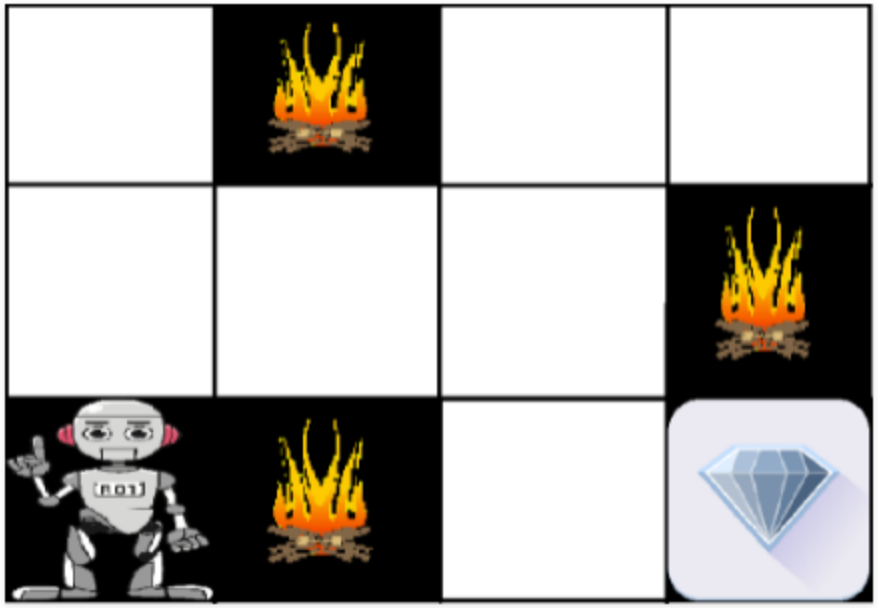


3 Reinforcement learning

- **Input**: The input should be an initial state from which the model will start
- **Output:** There are **many possible output** as there are variety of solution to a particular problem
- **Training**: The training is based upon the input, The model will return a state and the user will decide to reward or punish the model based on its output.
- The model keeps continues to learn.
- The best solution is decided based on the maximum reward.

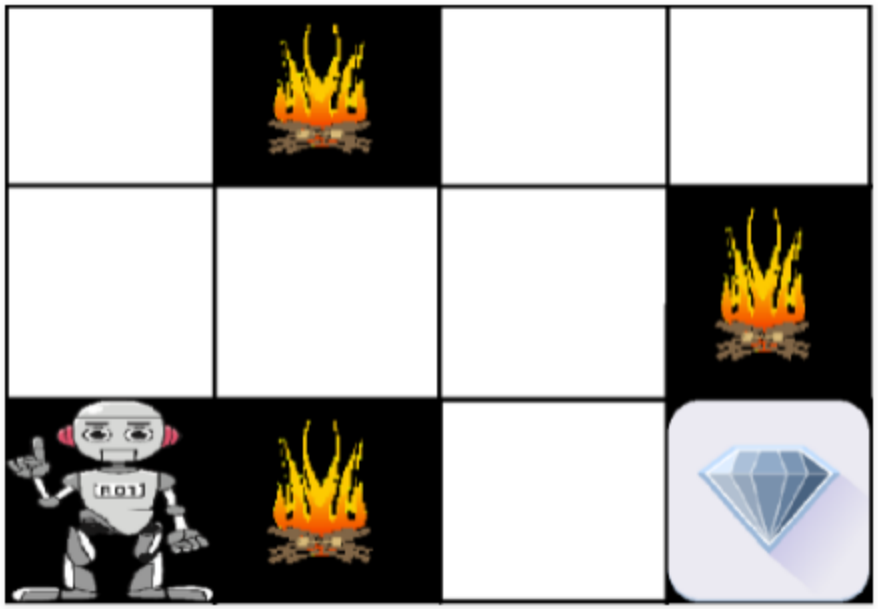
3 Reinforcement learning

- **Example:** The problem is as follows: We have an agent and a reward, with many hurdles in between. The agent is supposed to find the best possible path to reach the reward. The following problem explains the problem more easily.



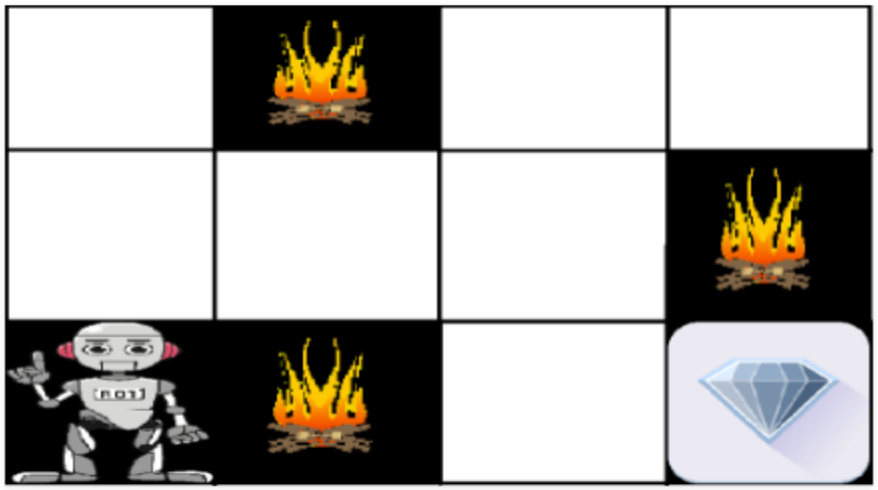
3 Reinforcement learning

- The image shows the robot, diamond, and fire. The goal of the robot **is to get the reward** that is the diamond and **avoid the hurdles that are fire.**



3 Reinforcement learning

The robot learns by trying all the possible paths and then choosing the path which gives him the reward with the least difficulty. Each right step will give the robot a reward and each wrong step will subtract the reward of the robot. The total reward will be calculated when it reaches the final reward that is the diamond.



Difference between Reinforcement learning and Supervised learning:

Reinforcement learning	Supervised learning
Reinforcement learning is all about making decisions sequentially. In simple words we can say that the output depends on the state of the current input and the next input depends on the output of the previous input	In Supervised learning the decision is made on the initial input or the input given at the start
In Reinforcement learning decision is dependent, So we give labels to sequences of dependent decisions	Supervised learning the decisions are independent of each other so labels are given to each decision.
Example: Chess game	Example: Object recognition

3 Reinforcement learning

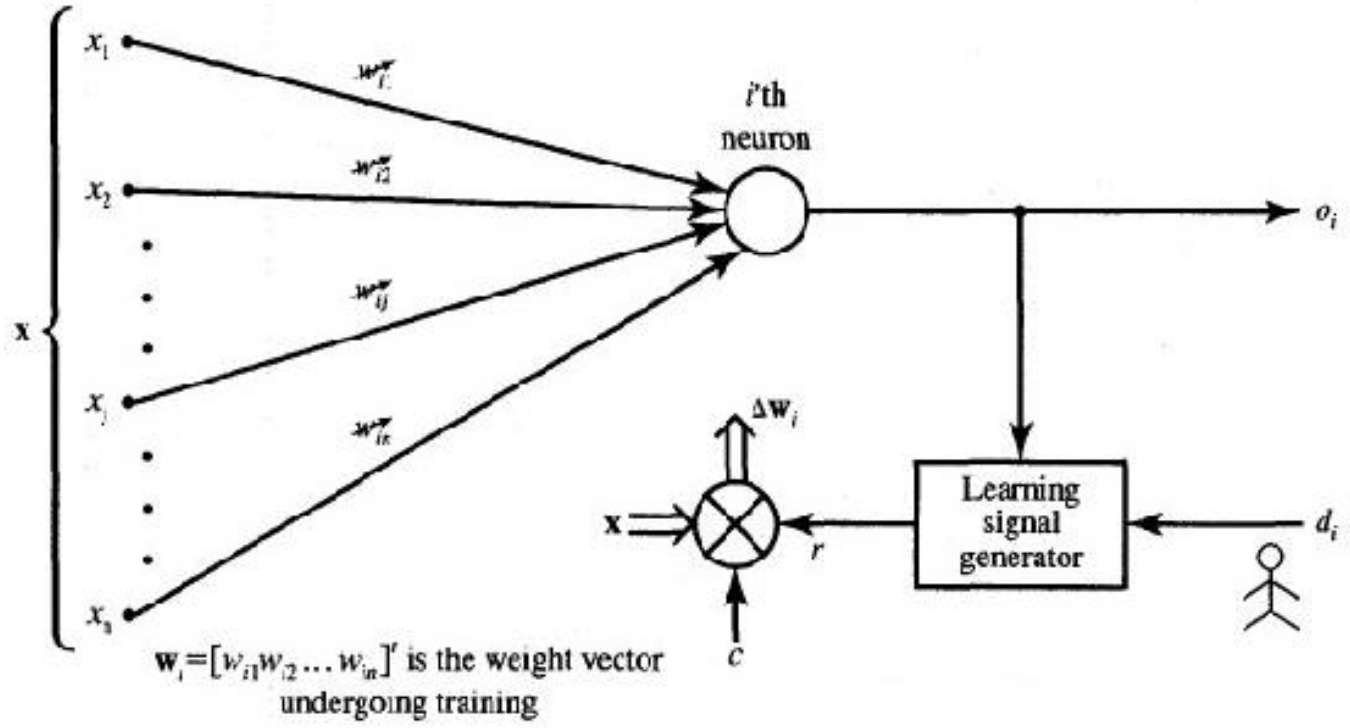
Various Practical applications of Reinforcement Learning –

- RL can be used in robotics for industrial automation.
- RL can be used in machine learning and data processing
- RL can be used to create training systems that provide custom instruction and materials according to the requirement of students.

Learning rule or Learning process

- Learning rule or Learning process is **a method or a mathematical logic**. It improves the Artificial Neural Network's performance and applies this rule over the network. Thus learning rules updates the weights and bias levels of a network when a network simulates in a specific data environment.

Learning rule or Learning process



1.3 Learning Rules:

- **Hebbian Learning Rule,**
- **Perceptron Learning Rule,**
- **Delta Learning Rule,**
- **Widrow-Hoff Learning Rule,**
- **Correlation Learning Rule,**
- **Winner Take-All Learning Rule.**

Hebbian Learning Rule

- **Hebbian Learning Rule**, also known as Hebb Learning Rule, was proposed by **Donald O Hebb.**
- It is one of the first and also easiest learning rules in the neural network.
- It is used for pattern classification. It is a single layer neural network,
- i.e. it has one input layer and one output layer. The input layer can have many units, say n.
- The output layer only has one unit.
- Hebbian rule works by **updating the weights between neurons in the neural network for each training sample.**

Hebbian Learning Rule Algorithm :

- Set all weights to zero, $w_i = 0$ for i=1 to n, and bias to zero.
- For each input vector, S(input vector) : t(target output pair), repeat steps 3-5.
-

$$w_i(new) = w_i(old) + x_i y$$

$$b(new) = b(old) + y$$

Hebbian Learning Rule Algorithm :

- Set activations for input units with the input vector $X_i = S_i$ for i = 1 to n.
- Set the corresponding output value to the output neuron, i.e. y = t.
- Update weight and bias by applying Hebb rule for all i = 1 to n:

$$w_i(new) = w_i(old) + x_i y$$

$$b(new) = b(old) + y$$

Hebbian Learning Rule Algorithm :

$$w_i(new) = w_i(old) + x_i y$$

$$b(new) = b(old) + y$$

Hebbian Learning Rule Algorithm :

Implementing AND Gate :

INPUT				TARGET	
	x_1	x_2	b		y
X_1	-1	-1	1	Y_1	-1
X_2	-1	1	1	Y_2	-1
X_3	1	-1	1	Y_3	-1
X_4	1	1	1	Y_4	1

Truth Table of AND Gate using bipolar sigmoidal function

Hebbian Learning Rule Algorithm

- There are 4 training samples, so there will be 4 iterations. Also, the activation function used here is Bipolar Sigmoidal Function so the range is [-1,1].
- **Step 1 :**
- Set weight and bias to zero, w = [0 0 0]^T and b = 0.
- **Step 2 :**
- Set input vector $X_i = S_i$ for i = 1 to 4.
- X_1 = [-1 -1 1]^T
- X_2 = [-1 1 1]^T
- X_3 = [1 -1 1]^T

Implementing AND Gate :

	INPUT			TARGET	
	x_1	x_2	b		y
X_1	-1	-1	1	Y_1	-1
X_2	-1	1	1	Y_2	-1
X_3	1	-1	1	Y_3	-1
X_4	1	1	1	Y_4	1

Truth Table of AND Gate using bipolar sigmoidal function

Hebbian Learning Rule Algorithm

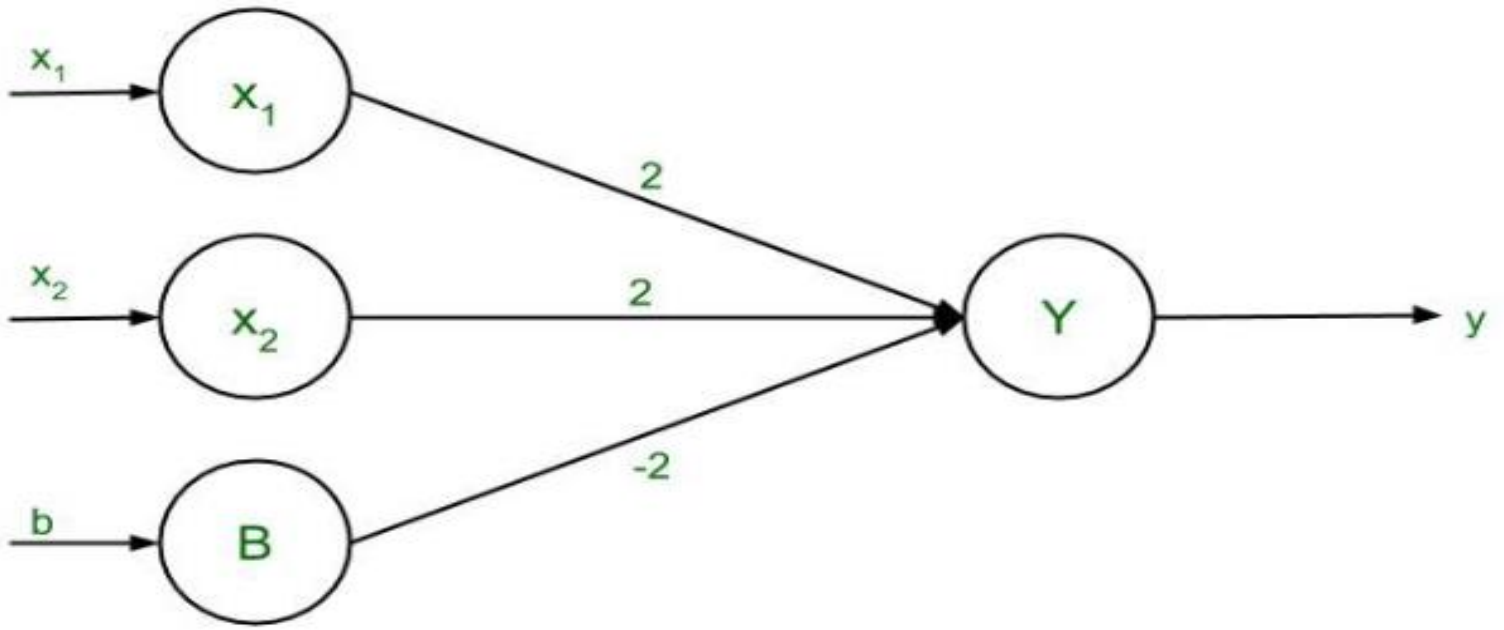
- $X_4 = [\ 1\ 1\ 1\ ]^T$
- **Step 3 :**
- Output value is set to y = t.
- **Step 4 :**
- Modifying weights using Hebbian Rule:
- First iteration –
- $w(new) = w(old) + x_1y_1 = [\ 0\ 0\ 0\ ]^T + [\ -1\ -1\ 1\ ]^T\ .\ [\ -1\ ] = [\ 1\ 1\ -1\ ]^T$
- For the second iteration, the final weight of the first one will be used and so on.

Hebbian Learning Rule Algorithm

- Second iteration –
- $w(new) = [\ 1\ 1\ -1\ ]^T + [\ -1\ 1\ 1\ ]^T\ .\ [\ -1\ ] = [\ 2\ 0\ -2\ ]^T$
- Third iteration –
- $w(new) = [\ 2\ 0\ -2]^T + [\ 1\ -1\ 1\ ]^T\ .\ [\ -1\ ] = [\ 1\ 1\ -3\ ]^T$
- Fourth iteration –
- $w(new) = [\ 1\ 1\ -3]^T + [\ 1\ 1\ 1\ ]^T\ .\ [\ 1\ ] = [\ 2\ 2\ -2\ ]^T$
- So, the final weight matrix is $[\ 2\ 2\ -2\ ]^T$

Hebbian Learning Rule Algorithm

Testing the network :



The network with the final weights

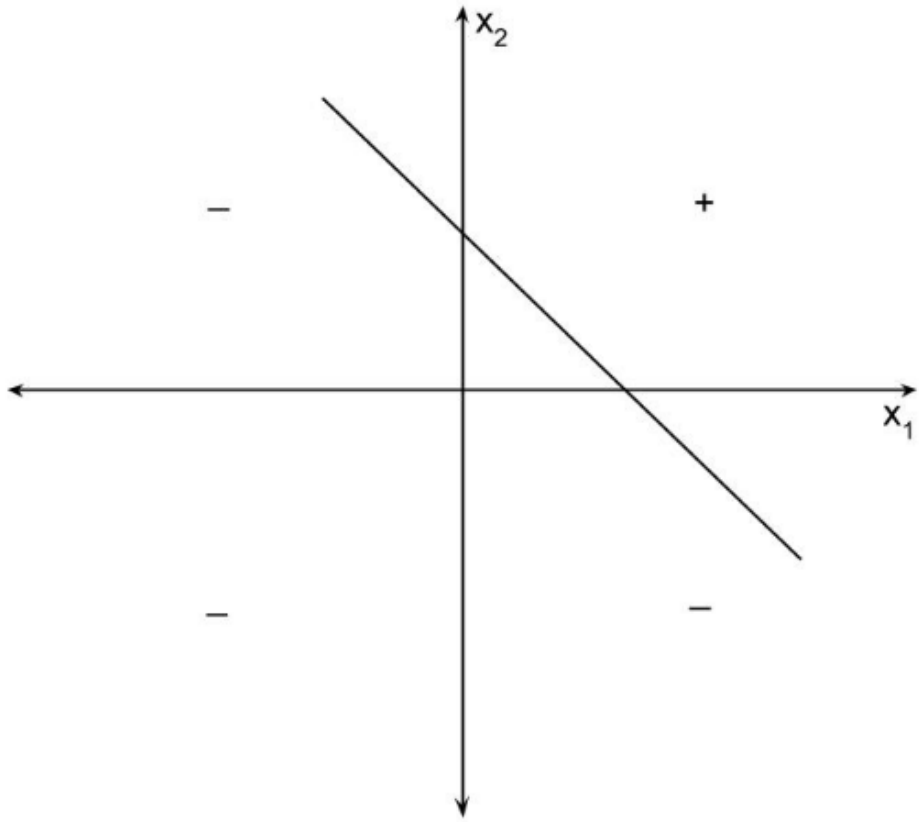
Hebbian Learning Rule Algorithm

- For $x_1 = -1$, $x_2 = -1$, b = 1, Y = (-1)(2) + (-1)(2) + (1)(-2) = -6
- For $x_1 = -1$, $x_2 = 1$, b = 1, Y = (-1)(2) + (1)(2) + (1)(-2) = -2
- For $x_1 = 1$, $x_2 = -1$, b = 1, Y = (1)(2) + (-1)(2) + (1)(-2) = -2
-

Hebbian Learning Rule Algorithm

- or $x_1 = 1$, $x_2 = 1$, b = 1, Y = (1)(2) + (1)(2) + (1)(-2) = 2
- The results are all compatible with the original table.
- **Decision Boundary :**
- $2x_1 + 2x_2 – 2b = y$
- Replacing y with 0, $2x_1 + 2x_2 – 2b = 0$
- Since bias, b = 1, so $2x_1 + 2x_2 – 2(1) = 0$
- $2(x_1 + x_2) = 2$
- The final equation, $x_2 = -x_1 + 1$
-

Hebbian Learning Rule Algorithm



Decision Boundary of AND Function

Perceptron Learning Rule

- If the output matches the target then no weight updation takes place.
- The weights are initially set to 0 or 1 and adjusted successively till an optimal solution is found.

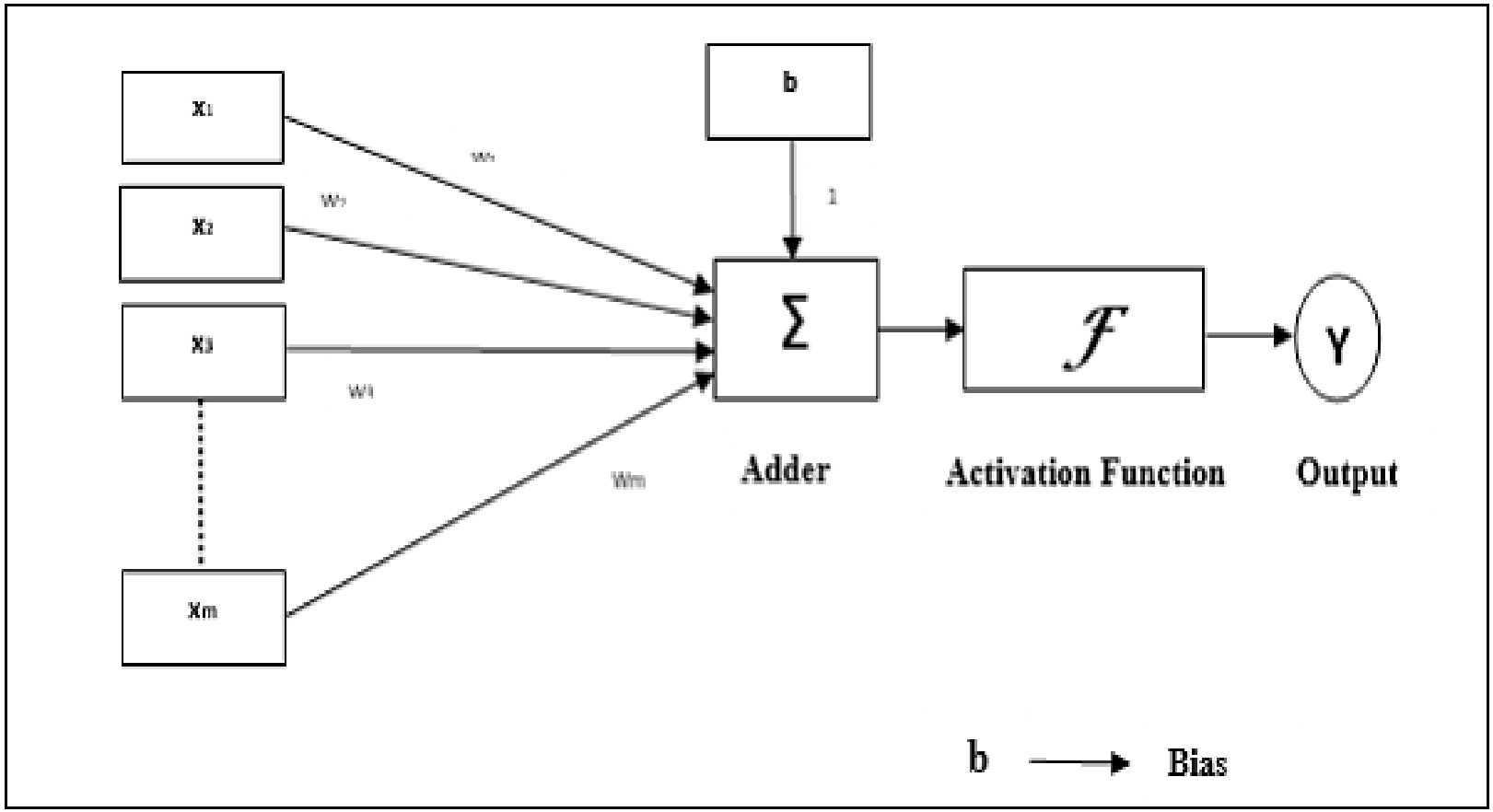
The weights are adjusted according to learning ruleα as:

$W(new) = w(old) + \alpha * t * x$

$B(new) = b(old) + \alpha * t$

α is the learning rate and t is the target.

Perceptron Learning Rule



Perceptron Learning Rule

- The error is calculated based on the actual output and the desired output.
- The weights in the network can be set to any values initially. The **Perceptron learning will converge to weight vector that gives correct output** for all input training pattern and this learning happens in a finite number of steps.
- The Perceptron rule can be used for both binary and bipolar inputs.

The weights are adjusted according to learning ruleα as:

$W(new) = w(old) + \alpha * t * x$

$B(new) = b(old) + \alpha * t$

α is the learning rate and t is the target.

Learning Rule for Single Output Perceptron

1) Let there be “n” training input vectors and x (n) and t (n) are associated with the target values.

2) . Initialize the following to start the training . Set them to zero for easy calculation

- Weights
- Bias
- Learning rate α (Let the learning rate be 1).

4) The input layer has identity activation function so x (i)= s (i).

Learning Rule for Single Output Perceptron

5) To calculate the output of the network:

$Y = b + \sum x * w$ for all input vectors from 1 to n.

6) The activation function is applied over the net input to obtain an output.

$$f(y_{in}) = \begin{cases} 1 & if\ y_{in} > \theta \\ 0 & if\ -\theta \leqslant y_{in} \leqslant \theta \\ -1 & if\ y_{in} < -\theta \end{cases}$$

Learning Rule for Single Output Perceptron

- **#7)** Now based on the output, compare the desired target value (t) and the actual output.

· Adjust the weight and bias as follows –

Case 1 – if **y ≠ t** then,

$$w_i(new) = w_i(old) + \alpha\, t x_i$$

$$b(new) = b(old) + \alpha t$$

Case 2 – if **y = t** then,

$$w_i(new) = w_i(old)$$

$$b(new) = b(old)$$

Here **'y'** is the actual output and **'t'** is the desired/target output.

Learning Rule for Single Output Perceptron

- **Step 8** – Test for the stopping condition, which would happen when there is no change in weight.

Perceptron learning Rule :NOT gate implementation

- https://colab.research.google.com/drive/1svLpQ0ndkmeqZFgiBkX4uN6vxdUsJZA4?usp=sharing

Example Of Perceptron Learning Rule

- **Implementation of AND function using a Perceptron network for bipolar inputs and output.**
- The input pattern will be x1, x2 and bias b.
- Let the initial weights be 0 and bias be 0.
- The threshold is set to zero and the learning rate is 1.

Example Of Perceptron Learning Rule

AND Gate

X1	X2	Target
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1

#1) X1=1 , X2= 1 and target output = 1
W1=w2=wb=0 and x1=x2=b=1, t=1
Net input= y =b + x1*w1+x2*w2 = 0+1*0 +1*0 =0

Example Of Perceptron Learning Rule

W1=w2=wb=0 and x1=x2=b=1, t=1

Net input= y =b + x1*w1+x2*w2 = 0+1*0 +1*0 =0

As threshold is zero therefore:

$$F(y) = \begin{cases} 1 \text{ if } y > 0 \\ 0 \text{ if } y = 0 \\ -1 \text{ if } y < 0 \end{cases}$$

From here we get, output = 0. Now check if output (y) = target (t).

y = 0 but t= 1 which means that these are not same, hence weight updation takes place.

Example Of Perceptron Learning Rule

$W(n) = w(o) + \alpha.t.x$

$W1(n) = 0 + 1 * 1 * 1 = 1$

$W2(n) = 0 + 1 * 1 * 1 = 1$

$B(n) = 0 + 1 * 1 = 1$

$\Delta w = \alpha.t.x$

$\Delta b = \alpha.t$

The new weights are 1, 1, and 1 after the first input vector is presented.

Example Of Perceptron Learning Rule

2) X1= 1 X2= -1 , b= 1 and target = -1, W1=1 ,W2=2, Wb=1

Net input= y =b + x1*w1+x2*w2 = 1+1*1 + (-1)*1 =1

The net output for input= 1 will be 1 from:

•

$$F(y) = \begin{cases} 1 \text{ if } y > 0 \\ 0 \text{ if } y = 0 \\ -1 \text{ if } y < 0 \end{cases}$$

Therefore again, target = -1 does not match with the actual output =1. Weight updates take place.

Example Of Perceptron Learning Rule

$W(n) = w(o) + \alpha.t.x$

$W1(n) = 1 + 1 * -1 * 1 = 0$

$W2(n) = 1 + 1 * -1 * -1 = 2$

$B(n) = 1 + 1 * -1 = 0$

$\Delta w = \alpha.t.x$

$\Delta b = \alpha.t$

Now new weights are w1 = 0 w2 =2 and wb =0

Similarly, by continuing with the next set of inputs, we get the following table:

Example Of Perceptron Learning Rule

- EPOCH1

Input		Bias	Target	Net Input	Calculated Output	Weight Changes			New Weights		
X1	X2	b	t	yin	Y	?w1	? w2	? b	W1	W2	wb
EPOCH 1											
1	1	1	1	0	0	1	1	1	1	1	1
1	-1	1	-1	1	1	-1	1	-1	0	2	0
-1	1	1	-1	2	1	1	-1	-1	1	1	-1
-1	-1	1	-1	-3	-1	0	0	0	1	1	-1

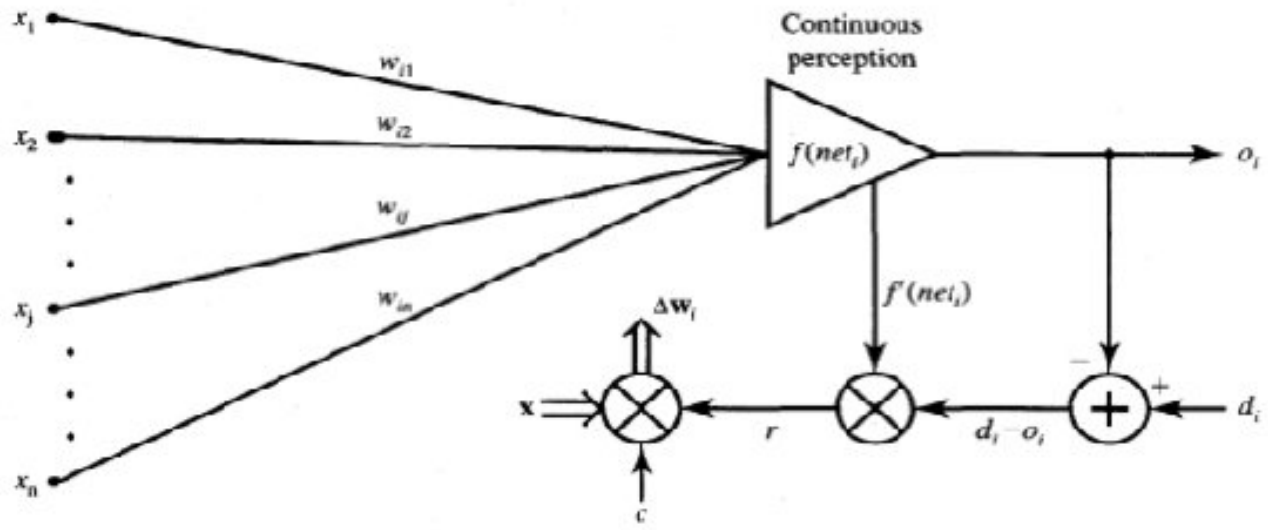
EPOCHS??????

- The EPOCHS are the cycle of input patterns fed to the system until there is no weight change required and the iteration stops.

EPOCH 2											
1	1	1	1	1	1	0	0	0	1	1	-1
1	-1	1	-1	-1	-1	0	0	0	1	1	-1
-1	1	1	-1	-1	-1	0	0	0	1	1	-1
-1	-1	1	-1	-3	-1	0	0	0	1	1	-1

Delta learning Rule

- Only valid for continuous activation function
- Used in supervised training mode
- Learning signal for this rule is called delta
- The aim of the delta rule is to minimize the error over all training patterns



Delta learning Rule

Delta learning Rule

- The delta learning rule only valid for continuous activation function and in the supervised training mode
- The learning signal for this mode is called delta and is defined as follows
-

$$r = [d_i - f(w_i^t \mathrm{x})] f'(w_i^t \mathrm{x})$$

the term $f'(w_i^t x)$ is the derivative of the activation function $f(net)$ computed for

$$net = w_i^t x$$

- Ref
- https://www.softwaretestinghelp.com/neural-network-learning-rules/